**Python Password Generator Documentation**

**Introduction**

Welcome to the Python Password Generator documentation in which I will share al the essential

Documentation for python password generator and clear instruction.

This document provides a detailed analysis of the code and describes its structure, functionality, and improvements.

Python Password Generator is a tool that allows you to create strong passwords using various combinations of letters, symbols, and numbers.

* **Python Password Generator**: This refers to a tool or script written in the Python programming language the file name called Python Password Generator.py .py is the extension of python program file.

**The purpose of this script is to generate a password.**

* **Strong passwords:** The passwords generated by this Python script are designed to be secure.

Security is achieved by including a specific combination of letters, symbols, and numbers in the generated password.

* **Customization:** Users can customize their passwords according to their needs.

Customization options typically include specifying the number of letters, symbols, and numbers to include in the generated password.

* **Ease of Use:** This script is described as easy to use and requires only a few steps.

The user is guided through the process through command line instructions.

* **Difficult to crack:** This script is intended to generate passwords that are difficult to crack or guess.

This is an important feature that improves the security of any account or system that uses these passwords.

* **Command line instructions:** The script will probably be executed via a command line interface.

Users interact with the script by entering commands on the command line, and the script responds by generating the required password.

* **Options:** Users have the option to choose between ``Ordinary password'' and ``Mixed password.

'' The term "Mixed" can refer to passwords in which the characters are rearranged in a random order, adding additional complexity.

In summary, this Python password generator is a tool that provides users with the ability to create strong, customizable passwords through a simple command line interface with options for a variety of password types.

For added security, emphasis is placed on creating strong, difficult-to-crack passwords.

**Password. Usage**

The instructions provided include instructions for generating passwords using a script or program.

**Let's take a closer look at each step below and provide you instructions:**

1. **Run a script:** This means running or starting a program (script) on your computer or device.

Depending on your platform, this may include double-clicking the file, typing commands in Terminal or a command prompt, or using certain software applications.

Please enter the number of letters, symbols and numbers: After running the script, the program will ask you to enter the required specifications for the password.

This typically involves specifying the number of letters, symbols, and numbers to include in your password.

**You may be asked to specify these as separate values.**

1. **Please select the type of password you require:** After entering the required details, you will be asked to select the type of password you require.
2. **There are two options available:**
3. **Ordinary Password:** This password type follows the order specified previously.

For example, if you require 5 letters, 2 symbols, and 3 numbers in that order, a typical password will follow that order.

1. **Mixed Passwords:** In contrast, mixed passwords randomly rearrange letters, symbols, and numbers and provide you mixed passwords.

As a result, the order becomes more random and the password becomes more complex.

In summary, this script helps you customize and generate passwords based on your preferences, allowing you to choose between a structured, common arrangement, or a more random mixed arrangement for added security.

Provides a flexible way to create passwords to suit your specific needs.

.

**Code Overview**

**Password Components**

This script is designed to generate a password by combining characters from four different lists: Characters: This list contains both lowercase and uppercase letters.

**For example,** you can include "a" through "z" for lowercase letters and "A" through "Z" for uppercase letters.

* **Number:** This list consists of numbers from 0 to 9.
* **Symbols:** This list contains frequently used symbols.

**Symbols can include characters such as "@", "#", "$", "%", "^", "&", and "\*"**

The script will likely use these lists to randomly select characters from each list and combine them to create a password.

Including characters from different categories (letters, numbers, and symbols) makes the generated passwords more secure and less predictable than those consisting of just one type of character.

Here's a simple Python example of how such a script would work.

**Python Copy code import Random Letters =**

'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ' Numbers = '01234 5 6789 ' Symbols = '@#$ % ^& \*' # Generate password by combining characters from list def generic\_password(length): password = [] for \_ in range(length): category = random.choice ([letters, numbers, symbols]) password.append(random.choice( category)) # Convert list of characters to string return ''.join(password) # Example: Generate a password of length 8 password = generic\_password(8) print( password) In this example, the generate password function randomly selects a category (letter, number, or symbol) for each character in the password, and then randomly selects a character in this category.

The resulting password will be a combination of characters from the specified list.

**User Input**

**This script is intended to generate passwords based on user input.**

The user will be asked to enter three pieces of information: **nr\_letters:** Number of characters to include in the password.

**nr\_symbols:** Number of symbols (!

, @, #, $, etc.

) to include in the password.

**nr\_numbers:** Number of numeric digits to include in the password.

The input() function is used to receive user input and int() is used to convert the input to an integer value.

This is because the number of letters, symbols, and numbers is assumed to be an integer.

For example, a user enters the following values: How many characters does a password need to be?

8 How many symbols does it need?

2 How many numbers do you want?

3 In this case, the user needs 8 characters, 2 symbols, We require a three-digit password.

The script can then use these values ​​to generate a password according to the user's settings.

The actual password generation logic is not included in the code snippet I shared.

This is implemented elsewhere in the script.

1. **Ordinary Password Generation**

The ordinary pass function generates a normal password by appending characters from each category in the specified order from the following code listed below.

**Python Code for Reference**

def ordinary\_pass():

    password = ""

    for characters in range(1, nr\_letters + 1):

        password += random.choice(letters)

    for charaters in range(1, nr\_symbols + 1):

        password += random.choice(symbols)

    for characters in range(1, nr\_numbers + 1):

        password += random.choice(numbers)

    print(f" First Ordinary Password is {password}")

1. **Mixed Password Generation**

The mixed pass function generates a shuffled password by first creating a list of characters from each category, shuffling the list, and then joining the characters into a string.

**Python Code for Reference**

def mixed\_pass():

    password\_list = []

    for characters in range(1, nr\_letters + 1):

        password\_list += random.choice(letters)

    for charaters in range(1, nr\_symbols + 1):

        password\_list += random.choice(symbols)

    for characters in range(1, nr\_numbers + 1):

        password\_list += random.choice(numbers)

    random.shuffle(password\_list)

    mixed\_pass = ""

    for characters in password\_list:

        mixed\_pass += characters

    print(f" Your Second Mixed Password is {mixed\_pass}")

* **User Interaction**

The script presents the user with options to choose the type of password generation.

**python**

while True:

    print("\n Types Of Password:")

    print("1. Ordinary Password ")

    print("2. Mixed Password")

selection = input("\nPlease enter the number of the action you want to perform: ")

    if selection == '1':

        ordinary\_pass()

    elif selection == '2':

        mixed\_pass ()

        break

    else:

        print("Please be sure the number you entered is valid.")

* **Error Handling**

If user write something wrong so this will show the error. (Kindly enter a positive number).

**python**

def get\_valid\_input(prompt):

    while True:

        try:

            value = int(input(prompt))

            if value < 0:

                raise ValueError("Kindly enter a positive number")

            return value

        except ValueError:

            print("Invalid input. Kindly enter a positive number")

print("Welcome to the Python Password Generator!")

nr\_letters = get\_valid\_input("The amount of characters do you want in your password?\n")

nr\_symbols = get\_valid\_input(f"Would you like a certain number of symbols?\n")

nr\_numbers = get\_valid\_input(f"What is your preferred number of digits?\n")

**Conclusion**

The "Conclusion" summarizes the key aspects and features of the Python Password Generator. Let's break down the key points step by step for further clear instruction and guidance:

**Purpose:**

The primary aim of the Python Password Generator is to create strong and unique passwords.

**Adjust to Your Preferences:**

Users can customize passwords based on their preferences, specifying the number of letters, symbols, and numbers in their password.

**Structured or Random Options:**

The generator provides flexibility with two options: "Structured Ordinary Password" and "Randomly Mixed Password." Users can choose between more predictable and structured passwords or more randomly arranged passwords.

**Focus on Safety:**

The script emphasizes the importance of staying safe, suggesting that the generated passwords are designed to enhance online security. This implies that the passwords generated are strong and unique, contributing to the user's overall safety in online activities.

**Versatility:**

The Python Password Generator is portrayed as a versatile tool that accommodates personal password creation preferences, offering both structured and random password options.

**Final Message:**

The conclusion ends with a final message urging users to prioritize online security. This serves as a call to action, encouraging users to recognize the significance of strong and unique passwords in safeguarding their online presence.

In summary, the Python Password Generator is presented as a versatile and user-friendly tool that empowers users to create secure and personalized passwords, emphasizing the importance of online security in the final message.

**MUHAMMAD SALMAN**

**0384610**